Homework 2 PDS — Answers

Ivan Aristy - iae225

1. Problem 1

1. Write a tuple relational calculus (TRC) query to find the ID and name of each student in the Comp. Sci. department

$$\begin{aligned}\{t|\exists s \in \text{student}(\\ t[\text{ID}] = s[\text{ID}] \wedge \\ t[\text{name}] = s[\text{name}] \wedge \\ s[\text{dept\_name}] = \text{“Comp. Sci.”})\}\end{aligned}$$

2. Write a relational algebra (RA) query to find the ID and name of each student in the Comp. Sci. department

$$\Pi_{\text{ID, name}}(\sigma_{\text{dept\_name}=\text{”Comp. Sci.”}}(\text{student}))$$

3. SQL

```
select ID, name from student
where dept_name = "Comp. Sci.";
```

2. Problem 2

I went to OH and the professor made me realize that ID is cs-101 but I assumed it was title. So for these questions onwards I was told to not change it and just leave the assumption as is. After all, it kinda made the questions more complicated, so she said it was ok.

4. Write a TRC query to find the ID of each instructor who has taught CS-101 along with the year in which they taught it.

$$\begin{aligned}\{t|\exists e \in \text{teaches}, s \in \text{section}, c \in \text{course}(\\ t[\text{ID}] = e[\text{ID}] \wedge \\ t[\text{year}] = e[\text{year}] \wedge \\ e[\text{course\_id}] = s[\text{course\_id}] \wedge \\ e[\text{sec\_id}] = s[\text{sec\_id}] \wedge \\ e[\text{semester}] = s[\text{semester}] \wedge \\ e[\text{year}] = s[\text{year}] \wedge \\ s[\text{course\_id}] = c[\text{course\_id}] \wedge \\ c[\text{title}] = \text{“CS-101”} \wedge \\ )\}\end{aligned}$$

Office hours edit:

$$\{t|\exists s \in \text{teaches}(\\ t[ID] = s[ID] \wedge \\ t[year] = s[year] \wedge \\ s[\text{course\_id}] = \text{CS-101})\}$$

5. Write an RA query to find the ID of each instructor who has taught CS-101 along with the year in which they taught it.

Assuming that the titile is cs101

$$\text{cs-101-courses} \leftarrow \sigma_{\text{title} = \text{“CS-101”}}(course)$$
$$\Pi_{\text{ID, year}}((\text{teaches}) \bowtie_{\text{teaches.course\_id} = \text{cs-101-courses.course\_id}} (\text{cs-101-courses}))$$

Office hours edit, going only from teaches:

$$\Pi_{ID,year}(\sigma_{\text{course\_id} = \text{”CS-101”}}(teaches))$$

6. SQL:

```
select
    ID,
    cs101courses.title as Title,
    year
from teaches
inner join (select * from course where course.course_id = "CS-101") as
cs101courses
on teaches.course_id = cs101courses.course_id;
```

3. Problem 3

I went to OH and the professor made me realize that ID is cs-101 but I assumed it was title. So for these questions onwards I was told to not change it and just leave the assumption as is. After all, it kinda made the questions more complicated, so she said it was ok.

7. Write a TRC query to find the ID and name of each instructor who has taught CS-101 along with the year in which they taught it.

$$\{t|\exists e \in \text{teaches}, s \in \text{section}, c \in \text{course}, i \in \text{instructor}(\\ t[\text{ID}] = e[\text{ID}] \wedge \\ t[\text{name}] = i[\text{name}] \wedge \\ t[\text{year}] = e[\text{year}] \wedge \\ e[\text{course\_id}] = s[\text{course\_id}] \wedge \\ e[\text{sec\_id}] = s[\text{sec\_id}] \wedge \\ e[\text{semester}] = s[\text{semester}] \wedge \\ e[\text{year}] = s[\text{year}] \wedge \\ e[\text{ID}] = i[\text{ID}] \wedge \\ s[\text{course\_id}] = c[\text{course\_id}] \wedge \\ c[\text{title}] = \text{“CS-101”} \wedge \\ )\}$$

8. Write a RA query to find the ID and name of each instructor who has taught CS-101 along with the year in which they taught it.

$$\text{cs-101-courses} \leftarrow \sigma_{\text{title} = \text{“CS-101”}}(course)$$
$$\text{instructor-teaches} \leftarrow (\text{instructor}) \bowtie_{\text{instructor.ID} = \text{teaches.ID}} (\text{teaches})$$
$$\Pi_{\text{id, year, name}}((\text{instructor-teaches}) \bowtie_{\text{instructor-teaches.course\_id} = \text{cs-101-courses.course\_id}} (\text{cs-101-courses}))$$

^ For the projection above pls assume the table names since I ran out of space to include them (ie. id = intructor-teaches.ID)

9. SQL

```
select
    teaches.ID,
    instructor.name as name,
    course.title as Title, # this is extra I know
    teaches.year as year
from teaches
inner join section on teaches.course_id = section.course_id
    and teaches.sec_id = section.sec_id
inner join course on section.course_id = course.course_id
inner join instructor on teaches.ID = instructor.ID
where course.course_id = 'CS-101';
```

4. Lecture 4 Material

10. Write an SQL query to find the total number of credits the student with ID 12345 has taken in Fall 2009. (Do not worry about whether they have a passing grade for the course.)

```
select
    sum(credits) as total_credits
from
    course
inner join takes on course.course_id = takes.course_id
where takes.ID = 12345 and
takes.semester = "Fall" and
takes.year = "2009";
```

11. Write an SQL query to find the ID and total number of credits taken by each student in Fall 2009. (Do not worry about whether they have a passing grade for the course.)

```
select
    takes.ID,
    sum(course.credits) as total_credits
from
    course
inner join takes on course.course_id = takes.course_id
where
    takes.semester = 'Fall' and
    takes.year = 2009
group by
    takes.ID;
```

12. Make up another question about the university data, write it in English, and write an SQL query to answer it. It should involve a join of at least two tables.

Write a SQL query to find the courses that student 12345 is authorized to take (no repeats, consider prerequisites). Try for 70557 as well.

```
create temporary table temp_courseWP as
select
    course.course_id,
    course.title,
    prereq.prereq_id
from
    course
left join prereq on course.course_id = prereq.course_id;

select distinct
    temp_courseWP.course_id,
    temp_courseWP.title
from
    temp_courseWP
where
    temp_courseWP.course_id not in (
    select takes.course_id
    from takes
    where takes.ID = '12345'
)
and
    (temp_courseWP.prereq_id is null or temp_courseWP.prereq_id in (
    select takes.course_id
    from takes
    where takes.ID = '12345'
    ))
;

select distinct
    temp_courseWP.course_id,
    temp_courseWP.title
from
    temp_courseWP
where
    temp_courseWP.course_id not in (
    select takes.course_id
    from takes
    where takes.ID = '70557'
)
and
    (temp_courseWP.prereq_id is null or temp_courseWP.prereq_id in (
    select takes.course_id
    from takes
    where takes.ID = '70557'
    ))
;
```

5. Retailer Database

13. Find the productCode, productName and productLine of each product ordered by any customer who lives in the USA that has status “shipped”

```
select
    #customers.customerName,
    #customers.country,
    products.productCode,
    products.productName,
    products.productLine
from orders
inner join customers on orders.customerNumber = customers.customerNumber
inner join orderdetails on orders.orderNumber = orderdetails.orderNumber
inner join products on orderdetails.productCode = products.productCode
where
    customers.country = "USA"
and
    orders.status = "shipped"
;
```

14. Find the total payments made by each customer who lives in the USA. The result should include the customer's customerNumber, customerName, and their total payments

```
select
    customers.customerNumber,
    customers.customerName,
    sum(payments.amount) as totalPayments
from
    payments
inner join customers on payments.customerNumber = customers.customerNumber
where
    customers.country = "USA"
group by
    customers.customerNumber,
    customers.customerName
;
```

15. For each productCode, list the productCode, productName, and the maximum profit on that product, i.e. the maximum difference between the buyPrice and the priceEach paid for ordered items of that product. You don't need to list products for which there were no orders.

```
select
    products.productCode as id,
    products.productName as product,
    max(orderdetails.priceEach - products.buyPrice) as maxProfit
from
    orderdetails
inner join
    products on orderdetails.productCode = products.productCode
group by
    products.productCode,
    products.productName
;

#previous iteration:
create temporary table productsales as
select
    orderdetails.productCode as id,
    products.productName as product,
    orderdetails.priceEach as pricePaid,
    products.buyPrice as basePrice
from
    orderdetails
inner join
    products on orderdetails.productCode = products.productCode
;

select * from productsales;

select
    id,
    product,
    max(pricePaid - basePrice) as maxProfit
from
    productsales
group by
    id,
    product
;
```